Using the nPM2100 Fuel Gauge

Application Note

4496_357 / 2025-07-01

Contents

Revision history

Date	Description
July 2025	Updated Battery model on page 6
March 2025	First release

1 Introduction

This document describes the seamless integration of the nPM2100 *Power Management Integrated Circuit (PMIC)* fuel gauge solution with a *System on Chip (SoC)* to estimate the battery state of charge for low-power device applications powered by non-rechargeable batteries.

The fuel gauge algorithm can run on any Arm® Cortex® M4 or M33 based SoC, including those found in nRF52 Series, nRF53 Series, and nRF54 Series SoCs. Other architectures can be supported based on customer requests.

The nPM2100 fuel gauge supports cylindrical alkaline AA and AAA typ) batteries and CR-type lithium coin cell batteries based on $LiMnO_2$ chemistry. Other battery types can be supported based on customer requests.

2 nPM2100 fuel gauge overview

The nPM2100 fuel gauge provides ultra-low power battery fuel gauging without compromising accuracy.

Estimating the state of charge for primary batteries, including alkaline AAA, AA, LR-series coin cells, and CR-series coin cells, is often challenging due to variations in battery quality, chemistry, and operating conditions. Traditional methods use lookup tables based on static voltage-to-state of charge mappings, which do not account for dynamic changes in battery behavior caused by varying loads and temperatures. These methods might also require external passive components to accurately measure battery voltage which add complexity to the design.

The nPM2100 fuel gauge is based on battery voltage and system temperature measurements. It features a fully integrated analog-to-digital converter (ADC) for real-time measurements accessible to the host *SoC* through a *Two-wire Interface (TWI)*.

The fuel gauge algorithm in the host SoC calculates the battery level as a percentage based on real-time measurements combined with a preconfigured generic model for each battery type which is developed from extensive testing across leading battery brands. This solution features built-in temperature compensation and provides more reliable battery charge estimates under varying load and environmental conditions.

The nPM2100 fuel gauge can easily be evaluated with an nPM2100 *Evaluation Kit (EK)* and the nPM PowerUP computer application.

The following figure shows an overview of the fuel gauge.

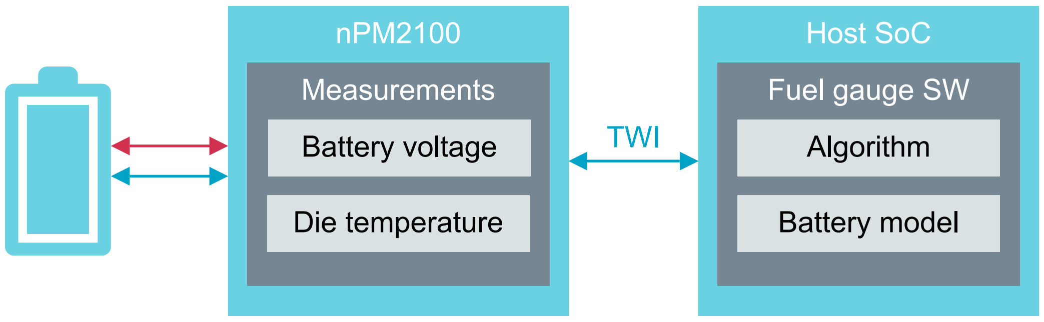

Figure 1: nPM2100 fuel gauge system

3 Battery model

The nPM2100 fuel gauge standardizes state-of-charge estimation for single-use primary cell batteries by averaging voltage and resistance which mitigates errors from manufacturing and chemical differences.

Unlike rechargeable batteries, primary cells are designed for single-use and are replaced multiple times throughout the device's life cycle. This variability in battery replacement introduces unique challenges, as primary batteries can differ significantly in performance and quality due to variations in manufacturing standards. Differences in battery chemistry, internal resistance, and build quality might lead to significant state-of-charge estimation errors if they are not accounted for.

The battery model of the nPM2100 fuel gauge represents the average voltage and resistance against state-of-charge relationships derived from extensive discharge cycle testing across various temperatures and multiple battery manufacturers for typical *Bluetooth*® Low Energy application loads.

The following battery models are currently available for implementation with the nPM2100 fuel gauge solution.

- AA Alkaline (single or two in series)
- AAA Alkaline (single or two in series)
- Alkaline coin cells
- $LiMnO_2$ based coin cells

The default battery model for alkaline coin cells is based on the LR44 battery. For $LiMnO_2$-based coin cells, the default battery model is based on the CR2032.

Other battery form factors and models might be supported. Contact Nordic Semiconductor for more information.

4 Fuel gauge configuration

Use the preconfigured battery model without modification or configure the fuel gauge settings to optimize performance for the end device application.

The nPM2100 fuel gauge provides configurable settings to optimize performance for various device applications. While the default settings are appropriate for most Bluetooth LE applications, developers can fine-tune the fuel gauge behavior as needed. For additional configuration options, see nrfxlib API: nRF Fuel gauge library.

4.1 Fuel gauge RAM retention

The fuel gauge library uses a few hundred bytes of memory for state management. The amount of memory needed depends on the fuel gauge version. For the best fuel gauge user experience, use RAM retention to ensure consistent and stable charge estimates. Without RAM retention, the memory must be reinitialized often which leads to less accurate predictions over time.

In applications using Ultra-Low Power (ULP) modes, such as nPM2100's Hibernate mode, where the host *SoC* is powered down, the RAM contents, including fuel gauge state memory, are lost. To preserve fuel gauge continuity, use nonvolatile memory options, such as RRAM or flash, to preserve the fuel gauge states before entering Low Power (LP) modes. Upon wakeup, the fuel gauge can resume operations seamlessly, avoiding the need for reinitialization.

Fuel gauge state memory can be retrieved by using the `nrf_fuel_gauge_state_get()` function. The application must ensure that this state information is stored properly. To resume operation from a stored state, use the `state` parameter in `nrf_fuel_gauge_init_parameters` when initializing the fuel gauge.

4.2 Fuel gauge initialization

Initialize the fuel gauge to accurately determine the initial state of charge and counteract the voltage recovery phenomenon in primary batteries.

If the host *SoC* reboots or starts from a fully OFF mode, the fuel gauge initializes based on initial battery voltage and system temperature measurements. Determining the initial state of charge accurately is challenging due to the voltage recovery phenomenon in primary batteries. This often results in an overestimated state of charge because the battery voltage temporarily rises during rest.

For the nPM2100 fuel gauge, initialize the fuel gauge during some load activity when battery current is above 2 mA to cause a dip in battery voltage to counteract the voltage recovery. This can be achieved by ensuring that the host SoC is active during the initial measurement or by briefly activating other system components, such as a status LED. Alternatively, using the boost in High Power (HP) mode to induce a high load above 2 mA for about 500 ms before fuel gauge initialization ensures a more realistic initial state-of-charge estimation.

Note: Fuel gauge should be initialized only once if the system starts from a fully OFF mode. After initialization, the fuel gauge operates continuously using real-time measurements to dynamically update the state of charge.

Reinitializing the fuel gauge upon wakeup from Bluetooth LE sleep or idle periods is unnecessary and should be avoided if RAM retention is enabled.

4.3 Average battery current configuration

Set the expected average battery current value during the active state to improve the reliability of the state-of-charge readings.

This can be determined by measuring the average current during normal device operation ahead of time. Provide the expected average current value to the `nrf_fuel_gauge_process()` function, just as you would with a measured real-time value.

For most device applications, the average current in the active state is deterministic. The average battery current value during active states can be configured to compensate for voltage drops by factoring in the average resistance values stored in the battery model. This allows the fuel gauge to provide more reliable state-of-charge readings even under varying load conditions.

For more information, see nrfxlib API: nRF Fuel gauge library and nPM2100 Fuel gauge sample.

4.4 Disable state-of-charge increase

The default setting of the fuel gauge allows the state of charge to increase during runtime. This behavior can be useful in cases where realistic state-of-charge increases occur, such as when the battery temperature fluctuates significantly over a short period of time. The fuel gauge can also be configured to prevent the state-of-charge values from increasing during runtime, which might enhance the user experience during continuous operation.

For more information, see nrfxlib API: nRF Fuel gauge library and nPM2100 Fuel gauge sample.

4.5 Battery level configurations and recommendations

The guidelines in this section describe how to interpret state-of-charge values effectively and integrate these into the end application.

Primary batteries, such as alkaline and $LiMnO_2$ cells, vary significantly in quality, voltage characteristics, and discharge behaviors due to differences in manufacturing standards and chemistries. These variations create challenges for state-of-charge estimation that uses a generic model and result in certain error margins in the fuel gauge readings.

By default, the fuel gauge outputs primary cell's state-of-charge values that are rounded off to appropriate thresholds. These thresholds are included in the battery model. The non-rounded state-of-charge value is also available as `soc_raw` field in the `nrf_fuel_gauge_state_info` structure.

The following table shows the recommended step sizes and typical error margins at different battery level ranges for an alkaline battery.

Battery level range (%)	Proposed step size	Typical error margins	Notes
100-90	5	±5	End users expect precise notifications. When a fresh battery is inserted, it should initialize the fuel gauge with a level close to 100%.
90-25	5	±10	End users are less sensitive to inaccuracies when the battery appears to be well charged.
25-0	5	±5	End users expect accurate battery level readings to avoid unexpected shutdowns, enable low-power state transitions, and prepare well ahead for battery replacement.

Table 1: Recommended fuel gauge step sizes at different battery levels for an alkaline battery

The following figure shows the typical error margins for different battery level ranges for an alkaline battery.

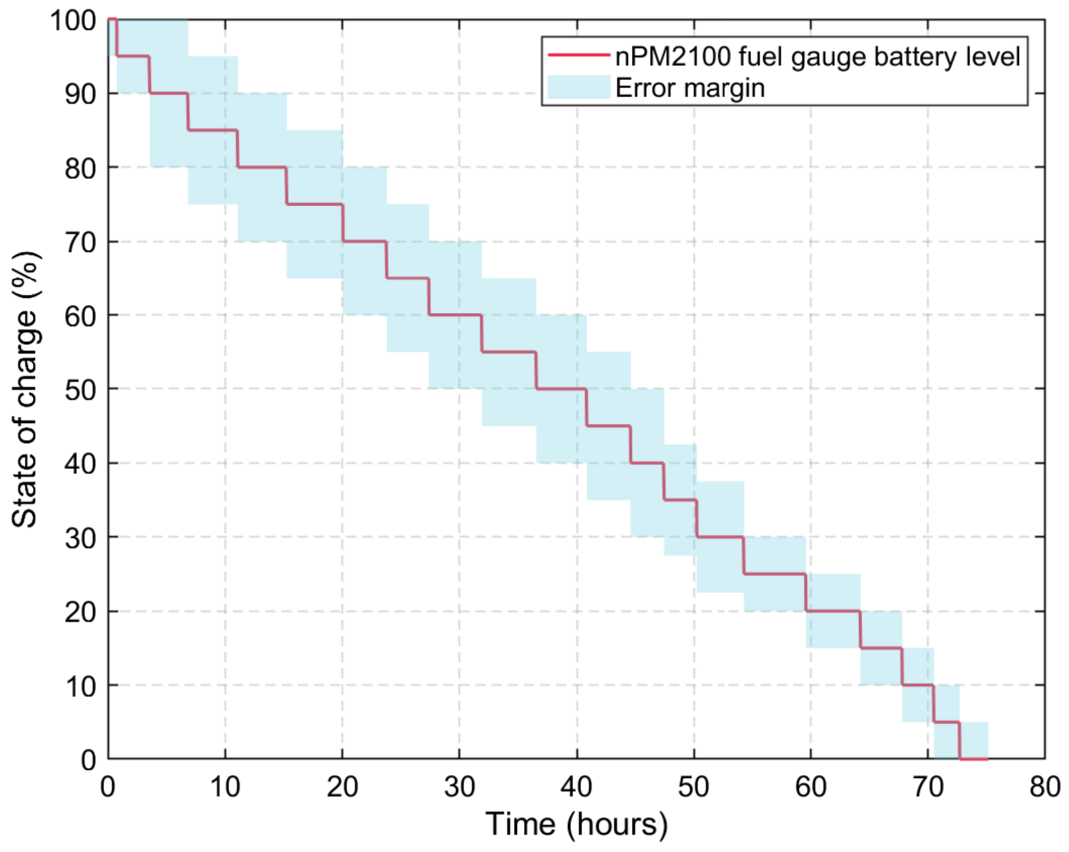

Figure 2: Battery level ranges with error margin for an alkaline battery

The following table shows the recommended step sizes and typical error margins at different battery level ranges for a CR-series ($LiMnO_2$) battery.

Battery level range (%)	Proposed step size	Typical error margins	Notes
100-75	25	±12.5	End users expect precise notifications. When a fresh battery is inserted, it should initialize the fuel gauge with a level close to 100%.
75-25	25	±25	End users are less sensitive to inaccuracies when the battery appears to be well charged.
25-0	5	±5	End users expect accurate battery level readings to avoid unexpected shutdowns, enable low-power state transitions, and prepare well ahead for battery replacement.

Table 2: Recommended fuel gauge step sizes at different battery levels for a CR-series ($LiMnO_2$) battery

The following figure shows the typical error margins for different battery level ranges for a CR-series ($LiMnO_2$) battery.

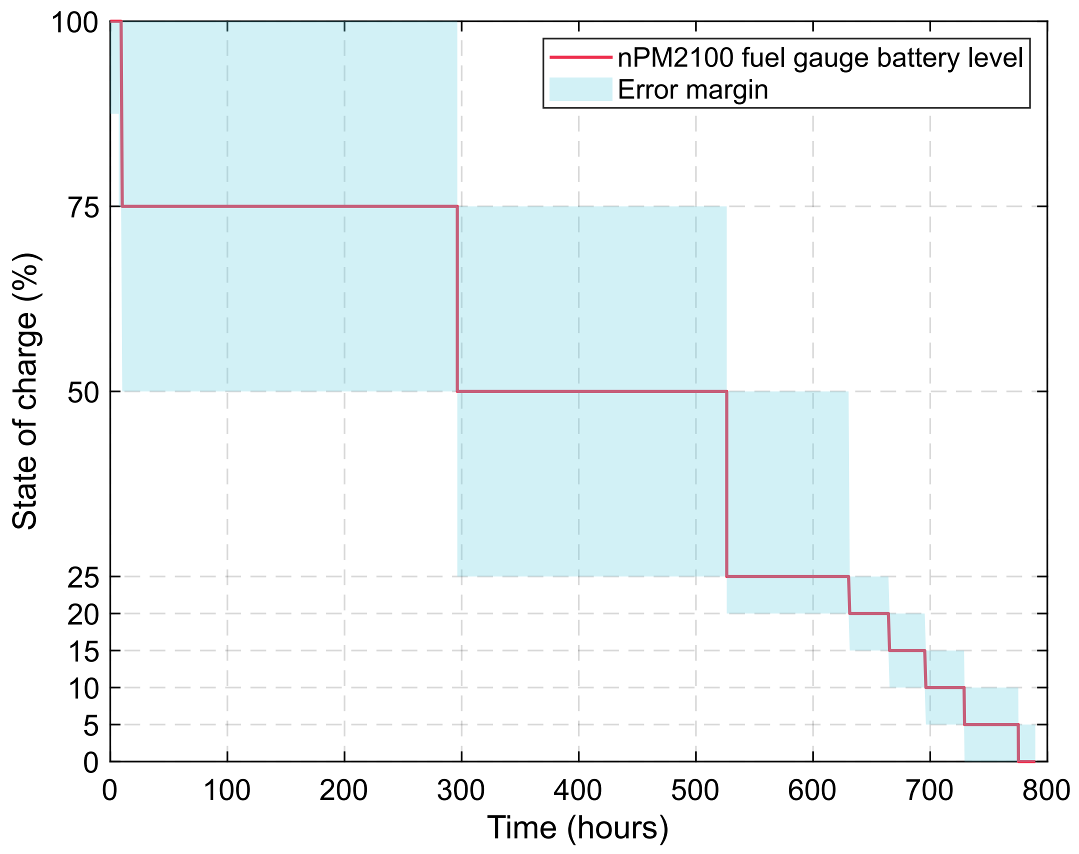

Figure 3: Battery level ranges with error margin for a CR-series ($LiMnO_2$) battery

The following figure shows examples of fuel gauging using recommended step sizes for an alkaline battery and a CR-series ($LiMnO_2$) battery.

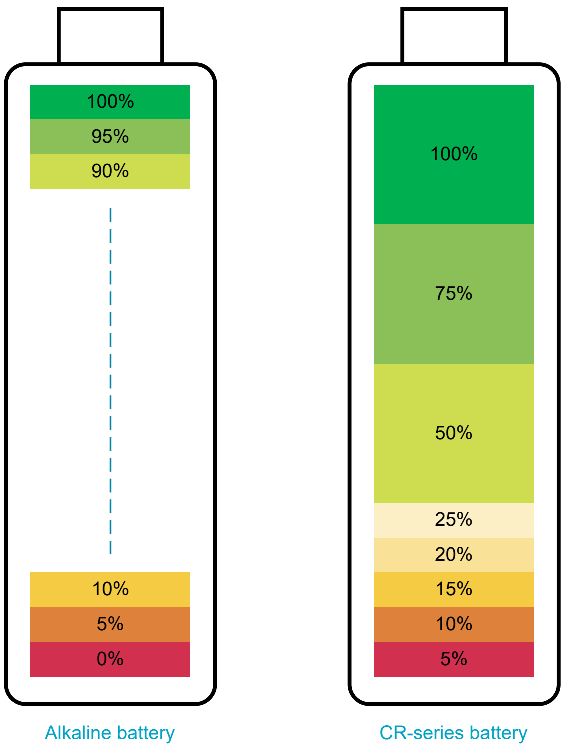

Figure 4: Fuel gauge example for alkaline and CR-series batteries

5 Performance plots

The estimated state of charge from the nPM2100 fuel gauge is compared with the measured data and estimation using the lookup table method.

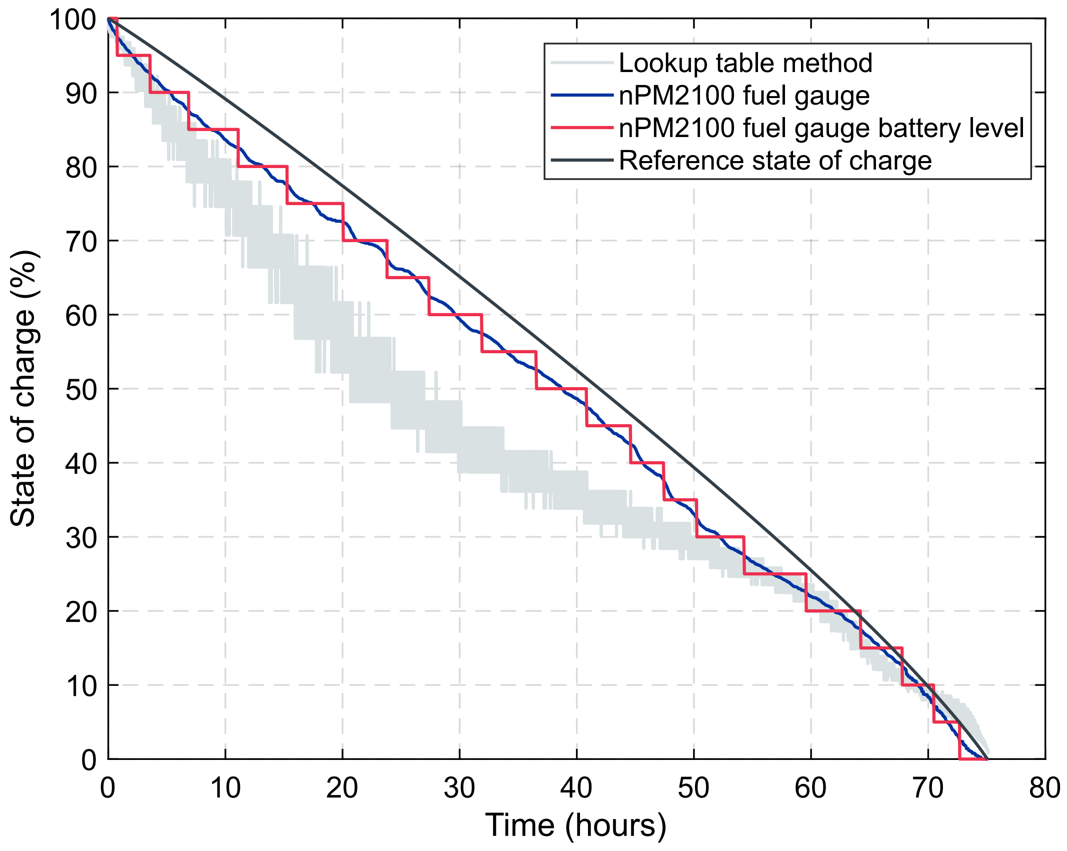

Figure 5: State of charge estimation for an alkaline battery

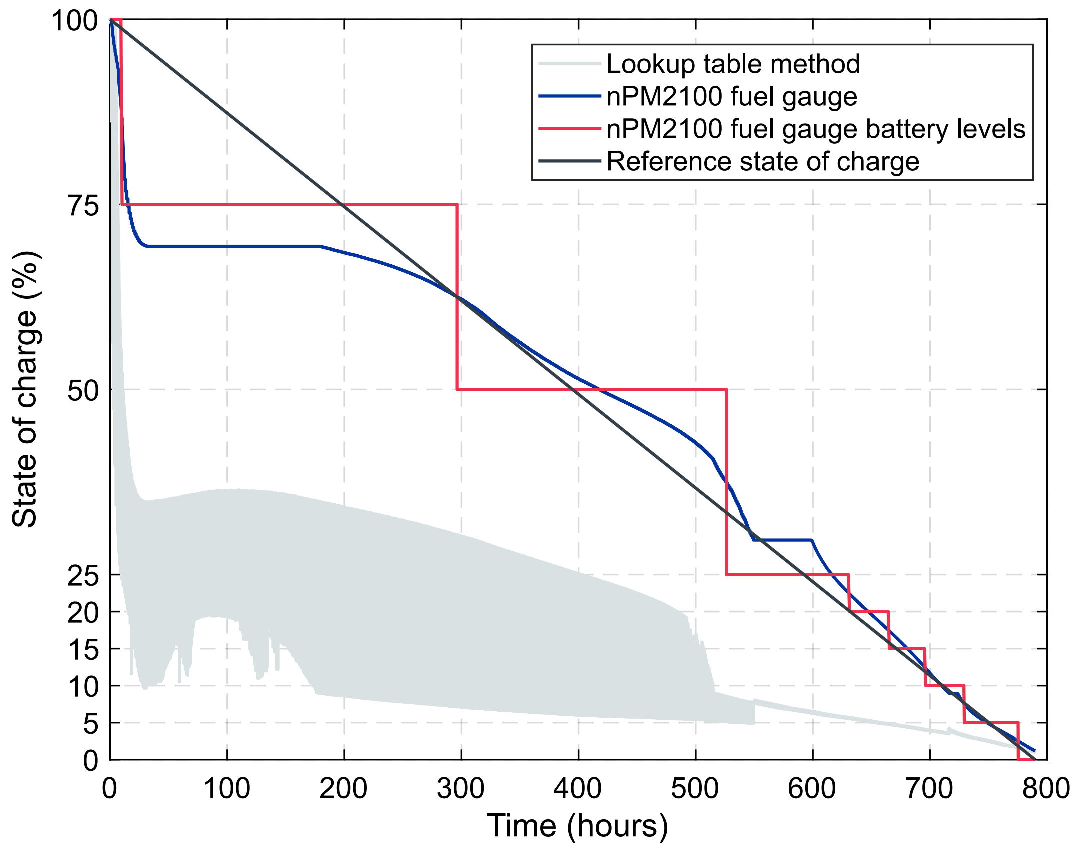

Figure 6: State of charge estimation for a CR2032 battery

6 Fuel gauge power consumption

The total current consumption of a battery fuel gauge is affected by several factors, such as operating mode, measuring frequency, *TWI* communication, and CPU usage. nPM2100 provides ultra-low power fuel gauging through an algorithm designed to operate only during active device periods, which eliminates the need for system wakeups.

The following table shows an example for fuel gauge current consumption and resource usage when using the nRF5340 *SoC* as the host.

Parameter	CPU usage	TWI communication	Measurements
Time consumed (ms)	0.1	2.4	5
Average current (mA)	3.6	1.0	0.17
Charge per iteration (µC)	0.4	2.4	1
Total charge for each fuel gauge iteration (µC)	3.9		

Table 3: Fuel gauge current consumption and resource usage example

The typical fuel gauge consumption for each fuel gauge iteration is less than 4 µC. This includes CPU usage, TWI communication, and measurements. It is recommended to run the fuel gauge at least once per second during the active state of the device.

The following table shows how fuel gauging adds extra overhead for a Bluetooth LE example application.

Device state	Time (s)	Average current (mA)	Single event charge (µC)	Fuel gauge average current (µA)	Fuel gauge event charge (µC)	Bluetooth LE event charge (µC)	Bluetooth LE event and fuel gauge event charge (µC)	Fuel gauge overhead (%)
Active	1	2	2000	4	4	2590	2594	0.15
Idle	59	0.01	590	0	No iterations			

Table 4: Fuel gauging power consumption and overhead example

The average current values used in the tables are from laboratory measured values and the nRF5340 Product Specification:

- nRF5340 application CPU current, running CoreMark® from flash at 64 MHz clock
- nRF5340 current consumption for TWIM, transferring data at 400 kbps

7 Fuel gauge sample

The nPM2100 fuel gauge sample demonstrates state-of-charge calculation for primary batteries using the nPM2100 *EK* and the nRF Fuel Gauge library to support early evaluations.

The sample utilizes nPM2100 *PMIC* drivers for implementation.

For more information, see nrfxlib API: nRF Fuel gauge library and nPM2100 Fuel gauge sample.

Glossary

Evaluation Kit (EK)

A platform used to evaluate different development platforms.

Power Management Integrated Circuit (PMIC)

A chip used for various functions related to power management.

System on Chip (SoC)

A microchip that integrates all the necessary electronic circuits and components of a computer or other electronic systems on a single integrated circuit.

Two-wire Interface (TWI)

An I^2C compatible serial communication protocol that enables devices to exchange data by using a two-wire bus system, allowing multiple devices to be connected and controlled by a master device.

Recommended reading

In addition to the information in this document, you may need to consult other documents.

Nordic documentation

- nPM2100 Datasheet
- nPM2100 EK Hardware
- nPM2100 EK product page

Legal notices

QUALITY SYSTEM CERTIFICATION
DNV
ISO 9001